Charger Active Defense v1.0 Team 2 - Group 12

## Team Lead in the Reporting Period: Noah Sickels

Noah Sickels, Adam Brannon, William Lochte

## Background / Abstract

Modern attack tools are highly efficient, while the cybersecurity industry struggles with active defense capabilities. Current passive defenses primarily focus on mitigating threats through means that can disrupt business operations or can be circumvented. Additionally, with the rapid advancement of artificial intelligence and prompt engineering, attackers can instantly generate various attack tools to target organizations.

The Charger Active Defense project focuses on developing a network-based fuzzing workflow to effectively and comprehensively test these known and AI/LLM-generated attack tools. It aims to identify responses that may cause the attacking application to crash or hang. Responses generated could be saved and sent back to the adversary through a Python replay service during a detected attack. Due to the multi-threading nature of many attack tools like Masscan and Medusa, effectively fuzzing these tools is complicated. If a fuzzed response leads to a crash or hang, it typically occurs in a thread separate from the attacking application's instance. This crashed thread may either be reported as a false positive or not counted at all. Because of this discovery, at the customer's request, we are pivoting towards thoroughly fuzzing generated attack tools from large language models.

Since we have successfully applied the fuzzing workflow to Masscan, we will also pursue the integration of ThreadSanitizer (TSan) to determine whether current fuzz testing properly crashes threads. The end goal of this workflow is to assess how we can broaden this method to apply to other attack tools, providing organizations with a strategy to defend their systems against an ever-evolving threat landscape.

## Current Project Status, Issues, & Short-Term Activities & Goals

This reporting period focused on developing the sending and receiving modules of the Python active defense service to finalize its functionality. Additionally, we performed unit testing for the user interface and logging modules of the Python service.

Our short-term goals for this period were:

* Develop Python service network receiver module.
* Develop Python service network sending module.

Issue during this period:

* None; N/A

For the next reporting period, our short-term goals are:

| Responsibility | Task |
| --- | --- |
| Noah Sickels | Unit testing on Python service & Gen-AI fuzzing workflow tool integration.   Finalize Gen-AI Comparison Report. |
| Adam Brannon | Unit testing on Python service & Gen-AI fuzzing workflow tool integration.  Finalize Gen-AI Comparison Report. |
| William Lochte | Unit testing on Python service & Gen-AI fuzzing workflow tool integration.  Finalize Gen-AI Comparison Report. |

## Milestone Status Summary

| Milestone / Task | Projected Due Date | Completion Date | Status | Notes |
| --- | --- | --- | --- | --- |
| Python Service Network Receiver Module Developed | 3/28 | 3/4 | Complete | Module completed during a prior reporting period. |
| Python Network Service Receiver Module Developed | 3/28 | 3/1 | Complete | Module completed during a prior reporting period. |

## Level of Effort / Individual Responsibility Record

| Member | Hours / Period | Total Hours / Spring |
| --- | --- | --- |
| Noah Sickels | 7 | 165 |
| Adam Brannon | 2 | 65 |
| William Lochte | 3 | 58 |
| Total | 12 | 288 |

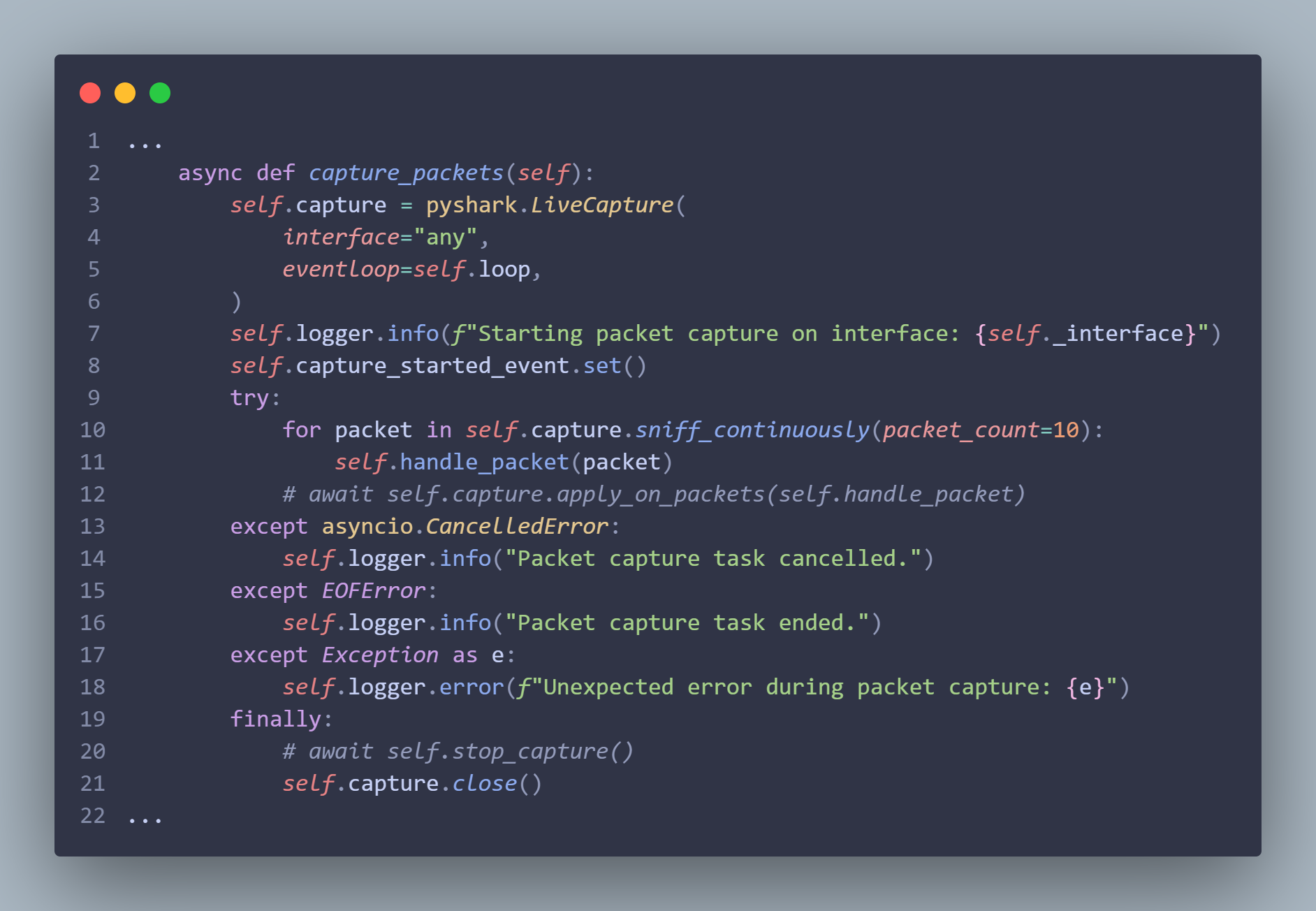
| Member | Individual Accomplishments |
| --- | --- |
| Noah Sickels | Performed unit testing on the logger module of the Python active defense service.  Performed unit testing on the various UI components of the Python active defense service. |
| Adam Brannon | Performed unit testing on the logger module of the Python active defense service.  Performed unit testing on the various UI components of the Python active defense service. |
| William Lochte | Performed unit testing on the logger module of the Python active defense service.  Performed unit testing on the various UI components of the Python active defense service. |

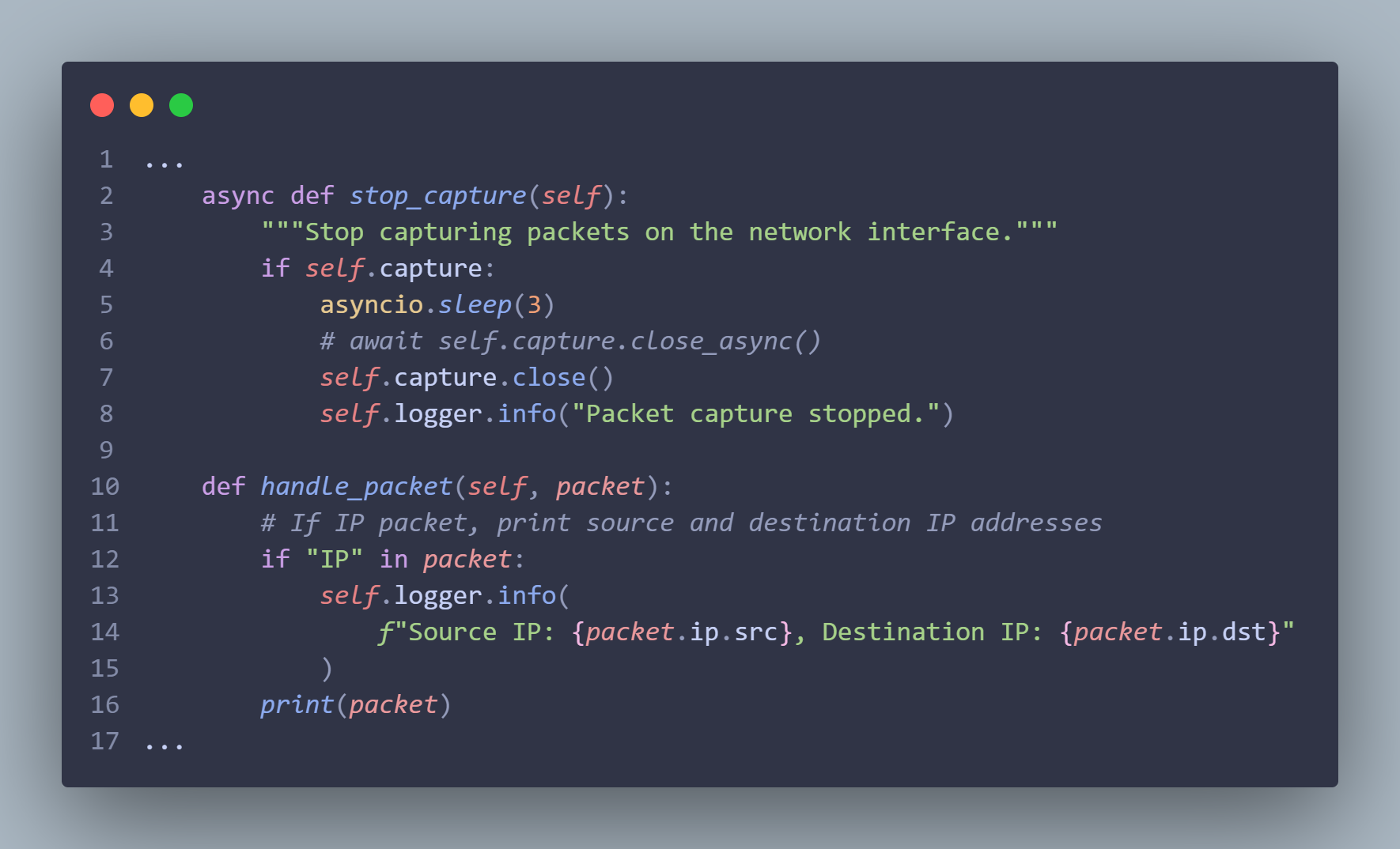
## Milestone Completion & Analysis

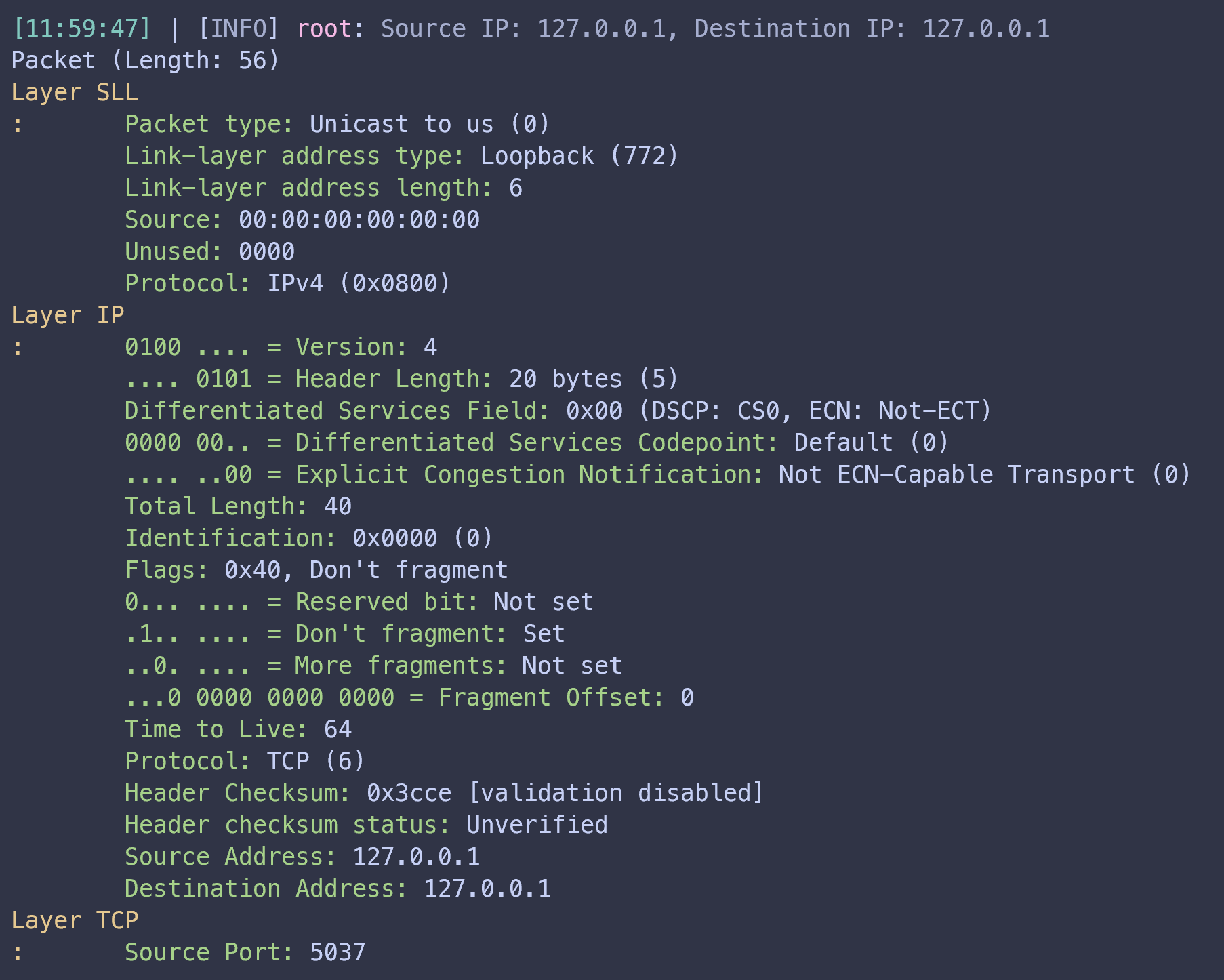
* Python service network traffic sending module developed.



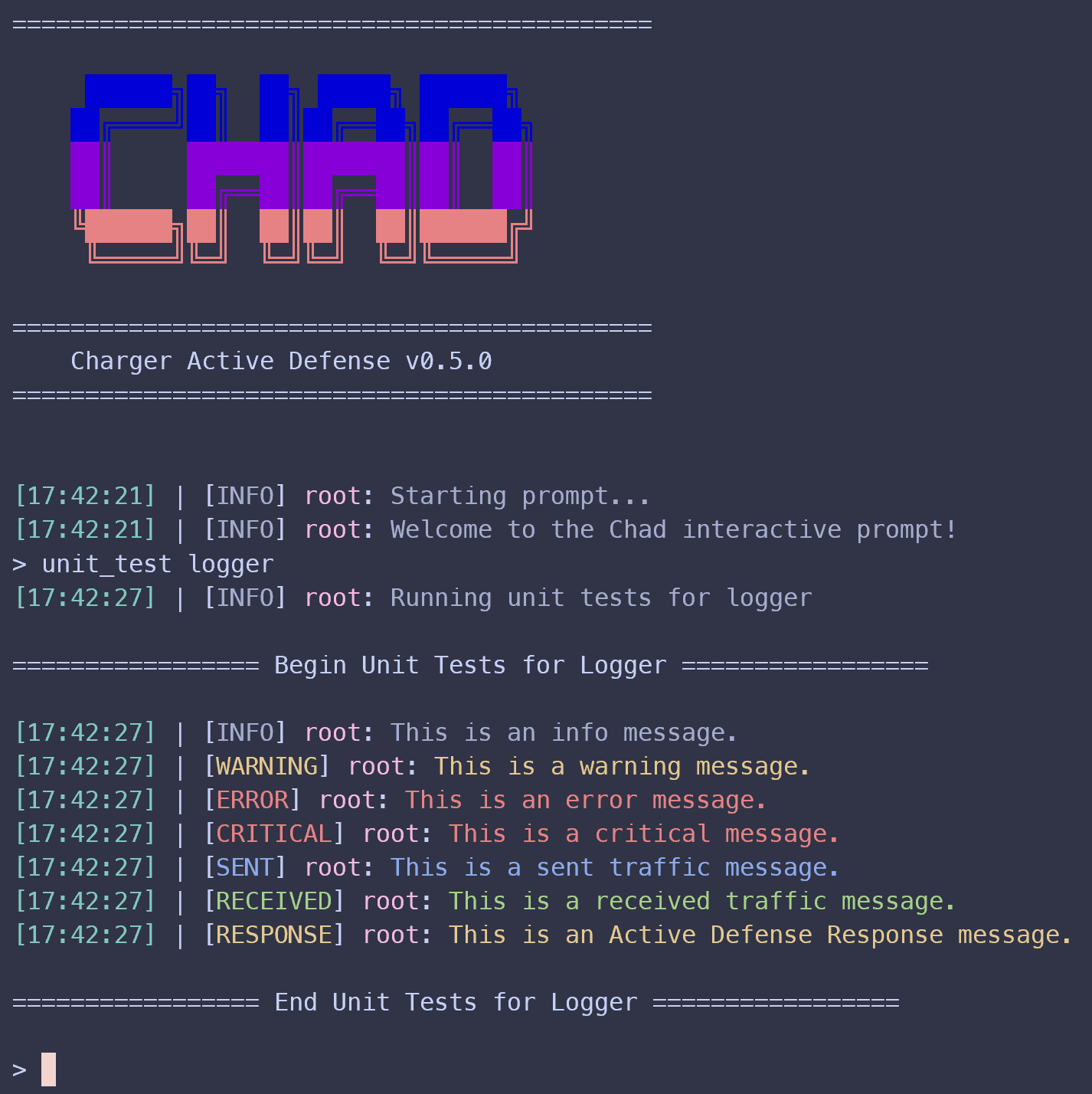
* Python service network traffic receiving module developed.







* Unit testing on the Python service logger module.



### Mitigations & Contingency Plans

* N/A - No missed milestones.